ViewCube: A 3D orientation indicator and controller

Conference Paper · January 2008

DOI: 10.1145/1342250.1342253 · Source: DBLP

5 authors, including:

Azam Khan
Autodesk Research
87 PUBLICATIONS **1,154** CITATIONS
SEE PROFILE

George Fitzmaurice
Autodesk Research
130 PUBLICATIONS **3,939** CITATIONS
SEE PROFILE

Justin Matejka
Autodesk Research
37 PUBLICATIONS **432** CITATIONS
SEE PROFILE

ViewCube: A 3D Orientation Indicator and Controller

Azam Khan, Igor Mordatch, George Fitzmaurice, Justin Matejka, Gordon Kurtenbach

Autodesk Research
210 King Street East, Toronto, Ontario, Canada
{firstname.lastname}@autodesk.com

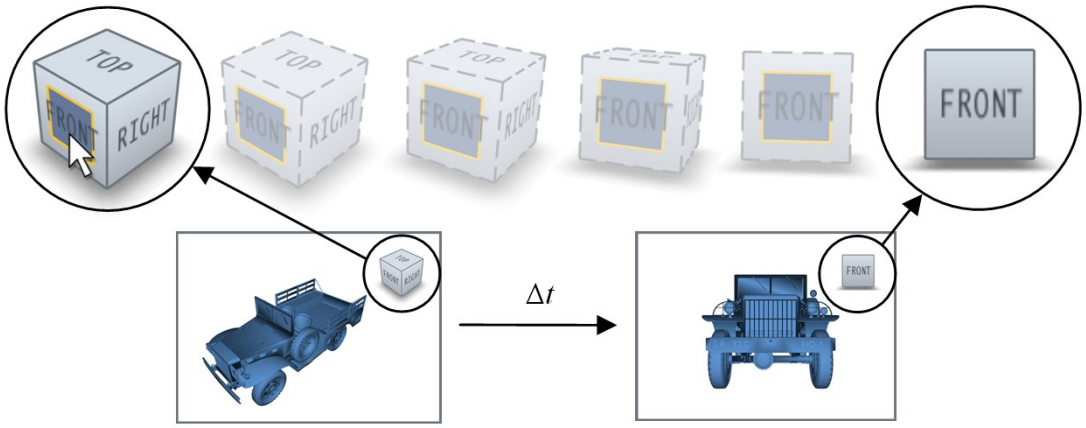

Figure 1. The ViewCube: clicking on the "front" face of the cube rotates the cube and the 3D scene to the front view.

Abstract

Literally hundreds of thousands of users of 2D computer-aided design (CAD) tools are in the difficult process of transitioning to 3D CAD tools. A common problem for these users is disorientation in the abstract virtual 3D environments that occur while developing new 3D scenes. To help address this problem, we present a novel in-scene 3D widget called the ViewCube as a 3D orientation indicator and controller. The ViewCube is a cube-shaped widget placed in a corner of the window. When acting as an orientation indicator, the ViewCube turns to reflect the current view direction as the user re-orients the scene using other tools. When used as an orientation controller, the ViewCube can be dragged, or the faces, edges, or corners can be clicked on, to easily orient the scene to the corresponding view. We conducted a formal experiment to measure the performance of the ViewCube comparing: (1) ArcBall-style dragging using the ViewCube for manual view switching, (2) clicking on face/edge/corner elements of the ViewCube for automated view switching and (3) clicking on a dedicated row of buttons for automated view switching. The results indicate that users prefer and are almost twice as fast at using the ViewCube with dragging compared to clicking techniques, independent of a number of ViewCube representations that we examined.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUI), 3D graphics

Additional Keywords and Phrases: 3D navigation, 3D widgets, Desktop 3D environments, virtual camera.

I3D 2008, Redwood City, California, February 15–17, 2008.
© 2008 ACM 978-1-59593-983-8/08/0002 $5.00

1 Introduction

Despite the addition of 3D modeling functionality to 2D drafting and computer-aided design (CAD) applications in the late 1990's, hundreds of thousands of users have been slow to move to 3D and so, have not been able to benefit from the many advantages of working in 3D. Through several exploratory usability studies[1] conducted with experienced users of 2D design applications as well as inexperienced users who needed to examine 3D data in the normal performance of their occupation, we discovered that the additional concepts and tools needed for orientation and navigation through a 3D scene were difficult for people to grasp.

Typical 3D applications allow users to create, manipulate, and view 3D geometry on traditional two-dimensional displays. By rendering a view of the virtual 3D scene from a particular viewpoint, a 2D image can be shown on the display. While this allows a rich and effective means of simulating the experience of viewing real 3D objects and scenes, controlling the virtual viewpoint and understanding the position of the viewpoint relative to the object is a significant task for users new to 3D. To help facilitate this paradigm transition, we developed a number of tools to address these issues. In particular, this paper focuses on the orientation issues.

1.1 Exploratory Study Observations

In observing inexperienced users and their first encounters with 3D applications, we found that many were confused and misinterpreted 3D scenes. For example, when first shown a perspective view of a box (with typical foreshortening), and are then shown a side view of the box (without any animated rotation between the two view), users misinterpreted the results. Typical comments would be: "my box disappeared and there's only a square now". Clearly for some users, they were confounded by

[1] Internal usability studies were conducted by our Usability Group.

their familiarity with 2D images from their drafting work and had a difficult time understanding and realizing that there were hidden elements in the depth dimension. Once they were told, or realized that the object was three dimensional, users requested different views of the object not by specific view naming conventions like "front view" or "bird's eye view", etc. but by spatial relationships relative to their current view. For the box example, participants stated that they wanted "to look at the side around the edge" or "go around to the back side."

In observing inexperienced 3D users, we found that many would attempt to move through 3D space using only the traditional 2D navigation tools: zoom and pan, even though tasks called for the use of other (3D specific) tools such as "look around", "orbit", or "fly/walk". Also, the first few minutes of use would often result in the viewpoint looking off into space where no data existed, or inside parts of the 3D model and/or upside down. In general, users reported a lack of understanding of their current orientation and how to go about correcting it once disoriented.

1.2 ViewCube

We present a new 3D widget called the ViewCube (see Figure 1) that is intended to address many of the orientation problems found in our exploratory study and in current methods. The ViewCube is a cube-shaped widget placed in the corner of the screen that serves as a proxy object for the 3D scene being created or modified. In the design of the widget, we apply the edicts of direct manipulation, namely "(1) continuous representation of the object of interest, (2) physical actions or labeled button presses instead of complex syntax, and (3) rapid incremental reversible operations whose impact on the object of interest is immediately visible". We discuss the design rationale behind the ViewCube and its features, and evaluate the performance of the ViewCube.

2 Related Work

2.1 3D Navigation

The most pervasive navigation metaphor is the cinematic camera model, enabling users to rotate, pan and zoom the viewpoint. Researchers have also explored other camera metaphors including "arcball" orbiting [Shoemake 1992] and flying [Stoev and Straber 2002], using constraints [Mackinlay et al. 1990; Singh and Balakrishnan 2004; Hanson and Wernet 1997], drawing a path [Igarashi et al. 1998], through-the-lens control [Gleicher and Witkin 1992], points and areas of interests [Jul and Furnas 1998], two-handed techniques [Balakrishnan and Kurtenbach 1999; Zeleznik and Forsberg 1999], and combinations of techniques [Shneiderman 1983; Smith et al. 2001; Ware and Osborne 1990]. Bowman et al. present taxonomies and evaluations of various interactions and camera models [1999; 1997]. A variety of metaphors have been explored for navigating in a virtual environment such as "eyeball", "scene in hand", and "flying" [Ware and Fleet 1997]. Finding effective viewing angles to facilitate 3D direct manipulation has also been investigated [Phillips et al. 1992]. Techniques such as HoverCam orbit objects at a constant distance and always face the object's surface [Khan et al. 2005]. Other navigation techniques include UniCam [Zeleznik and Forsberg 1999] click-to-focus [Mackinlay et al. 1990] and Tan et al. [2001].

2.2 View Switching

A room metaphor for managing multiple workspaces has also been introduced [Henderson and Card 1986; Chapuis and Roussel 2005]. Toolspaces [Pierce et al. 1999] are storage spaces attached to the user's virtual body where objects or views can be placed that are always accessible yet out of the user's view until needed. Users access these toolspaces to store and retrieve objects through a lightweight but rapid navigational glance.

With the ExoVis technique, a 3D object is surrounded by 2D orthographic views (e.g., front, top, left) which are projected onto planes surrounding the 3D object. Thus multiple 2D and the 3D perspective views are available simultaneously and within the same 3D space for improved view integration [Tan et al. 2001; Tory 2003; Tory and Swindells 2003].

Early 3D VRML environments [Rezzonico and Thalmann 1996] offer "3D bookmarks". The animated transitions between bookmark views, known as object constancy [Robertson et al. 1989], reduces the need to re-assimilate the material once at the new view because the user saw "how they got there", hence assisting orientation tasks.

To facilitate the realization that objects have depth, Grossman et al. [2001] introduced animated rotation transitions between the different views in the VRML bookmarks fashion. If the current and target views were not both using the same view projection, the projection matrix was manipulated to more smoothly transition between orthographic and perspective views.

The use of in-scene 3D widgets within the 3D workspace can facilitate 3D navigation tasks. Selectable "3D arrows" [Chittaro and Burigat 2004] point to objects of interest in the scene and, if selected, take the user to the target. The 3D arrows update and re-orientate as the user moves through the space or if the target moves. Within GeoZui3D [Komerska and Ware 2003] a "scene navigation widget" provides a 3D manipulator for the user with the ability to translate, rotate and scale the 3D scene about the center of the workspace.

The most closely related academic work is miniature overviews—"Worlds in Miniature" (or WIM) [Stoakley et al. 1995]. A WIM approach is directed towards immersive head tracked display environments and augments the display of 3D scene with a miniature copy of the scene which the user can "hold in their hand" and they control their viewpoint on the 3D scene by selecting locations on the miniature copy. Our solution differs in that we are, first, concerned with desktop virtual environments operated with standard mouse and keyboard and, second, instead of a literal miniature duplicate of the 3D scene, we use an abstract cube volume to represent and control viewing orientations.

2.3 Commercial Tools

In the commercial 3D application Maya, an in-scene widget called the ViewCompass provides more direct viewpoint selection, effectively replacing a small pull-down menu (see Figure 3). The ViewCompass is composed of a small cube in the scene that represents the current orientation of the scene, along with six cones pointing toward the faces of the cube. By clicking on the cone-shaped buttons, spatially associated with the faces of the cube, the virtual camera will turn to view that face, and both the model and the widget turn together during the animated transition. When clicking directly on the cube part of the ViewCompass, the camera moves to the standard three-quarter view (see top-right of Figure 2) regardless of which cube face was clicked. Users reported that this "return to the standard view" functionality was very useful as an orientation recovery mechanism when they

became disoriented in the 3D scene. ArcBall style dragging is not supported on the ViewCompass.

Maya's ViewCompass works towards a single-window workflow, avoiding the standard four-view approach by providing six standard orthographic views and one perspective view. However, we discovered that both engineering and architecture users have additional needs. For these classes of users, so-called "half views" (e.g. Front-Right) and additional "three-quarter views" (e.g. Top-Right-Back) are fundamental to their workflow. As such, in typical 3D engineering applications, the standard-views menu is significantly longer, often with a dozen views from which to choose. Including all of the "half" and "three-quarter" views with the six standard "face" views introduces an unwieldy total of twenty-six views.

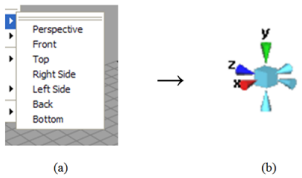

Figure 2. Standard Views (a) are accessible on the ViewCompass (b) where users can click the cones to view the scene from that point of view. Clicking the cube itself would move the camera to the standard three-quarter view with a perspective projection.

Autodesk's Inventor product, designed for CAD users, has an in-scene widget, called the "Glass Cube" (see Figure 3), that adds eight more viewpoints beyond the ViewCompass. Faces can be selected (by clicking on the green arrows on the faces) to move to that viewpoint. The corners of the Glass Cube can also be clicked on to move the viewpoint to a three-quarter view, providing a total of fourteen views. When the Glass Cube is "face on", clicking on a cube edge will roll the view of the model about the center of the closest face. The edge control is split so that clicking on one-half of an edge will roll clockwise and the other half will roll counter-clockwise. As such, edge-viewpoints cannot be selected. The green arrows on the back sides of the cube are selectable as are back roll edge segments. The Glass Cube fills a large part of the center of the screen, and as it is semi-transparent, is effectively in front of the object or scene. Users must invoke the widget by going to the arcball tool and selecting the Glass Cube mode via a hotkey. Once they have oriented the scene as desired, selecting a different tool will remove the Glass Cube. As this modal tool is not visibly available in the GUI, it is unclear how many users have discovered and used this feature. Finally, the Glass Cube controls only support clicking and do not support dragging.

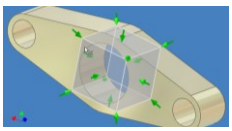

Figure 3. The Glass Cube has a complex set of controls: 14 selectable viewpoints (green arrows) and when "face on" edge segments are selectable and roll the view clockwise or counter-clockwise.

A number of factors make the Glass Cube objectionable even for the advanced users that we consulted. First, clicking through the transparent cube requires significant planning for the user to predict the outcome. Second, the roll operation is difficult to discover and operate. Finally, the Glass Cube itself visually interferes with the 3D scene being manipulated.

3 The ViewCube

To address all of the problems and user requirements stated above, we developed the ViewCube: a cube shaped widget with selectable pieces that spatially arranges the twenty-six possible views (8 corners, 12 edges, 6 faces – see Figure 4). The design is an extension of both the ViewCompass and of the Glass Cube but simplifies some interactions and makes some hidden functions explicit. Again, direct manipulation edicts are followed to assist novice 3D users.

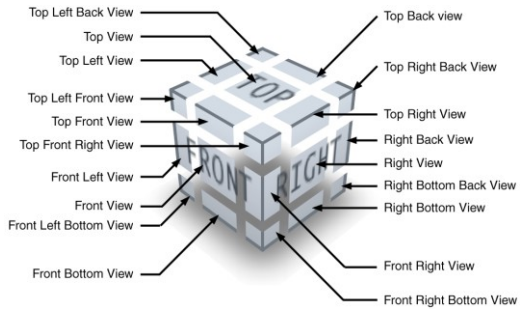

Figure 4. Split view diagram of the ViewCube from a ¾ view explicitly showing all currently selectable views (in this case 19 views are accessible).

The ViewCube maps the selectable views to a cube where the faces, edges and corners can be chosen (see Figure 4). These separate regions are effectively buttons where clicking a piece activates the command to set the viewpoint appropriately. Figure 1 shows the effect of clicking the Front button on the cube: the view of the model of the car and the view of the ViewCube both animate (over 0.5 seconds) until the front of the car is facing the user.

When inactive, the cube appears to be all one piece. So, to visually indicate which pieces of the cube are selectable, when the mouse pointer is above the cube, we highlight the part that would be selected if clicked (see Figure 5).

Figure 5. ViewCube Selection Feedback. As the cursor moves over the ViewCube, the piece that would be selected if the user clicked the mouse button is highlighted.

3.1 Dragging

When users click on a piece of the cube to select that view, a quick animated rotation moves the scene and the cube to that view (see Figure 1). But since the total hit area of the cube is quite large, it also affords dragging, and in a pilot study where only

clicking was provided, several users requested dragging. When the user clicks anywhere on the cube and begins to drag, the cube is rotated, along with the scene, providing freeform arcball-style direct manipulation of the camera position. Alternatively, a hotkey can be used to begin ViewCube dragging (for experienced users).

During freeform orientation of the view while dragging on the ViewCube, it can be very difficult to move exactly to one of the fixed views. As the cube's purpose is to select a fixed view, we added view snapping. While being dragged, when the viewpoint is within 10° of any of the fixed viewpoints, we snap to that view using snap-and-go [Baudisch et al. 2005; Bier 1990]. Due to the often abstract visual nature of the 3D scene under review (e.g. an incomplete wireframe display of a structure), we attempt to help the user easily determine if they are currently exactly at one of the fixed twenty-six views by drawing the outline of the cube using a solid line instead of a dashed line (see Figure 6).

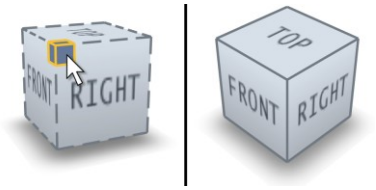

Figure 6. Before (left) and after (right): The dashed outline becomes solid to indicate that the camera is exactly at one of the fixed viewpoints.

When the user engages the application's orbit function to rotate the view of the scene, the ViewCube rotates as well to match. For additional feedback to the user, the piece of the ViewCube that represents the fixed viewpoint that is closest to the current scene orientation will be highlighted (see Figure 7). In this way, the user can easily see which piece of the cube to click on to quickly go exactly to the closest view, effectively to "repair" or align the current viewpoint. Furthermore, the "closest view" feedback makes it easier to choose a fixed view neighboring the current view to more easily make relative viewpoint changes. For example, clicking on a piece of the cube that is a direct (edge-to-edge) neighbor to the current viewpoint will result in a small viewpoint change (45°) while clicking a piece far from the current viewpoint will result in a much larger turn (up to 180° if going from the Top Front Right to the Top Left Back, for example). Without the "closest view" feedback, it is more difficult to judge how much of a viewpoint change will occur.

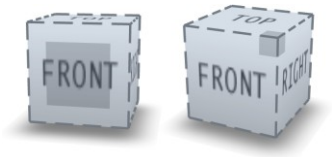

Figure 7. When moving the viewpoint using the application's Orbit tool, the cube rotates to match the scene orientation. The current closest fixed view is highlighted. Here, the scene is orbited slightly from almost Front (left) to almost Front-Top-Right (right).

Layout of Controls: 3D View-sensitive versus 2D Fixed

In Figure 2(a), we see a list of fixed viewpoints presented in a menu. This could be called a "2D fixed layout" while the spatial arrangement of the viewpoint buttons of the ViewCube design can be thought of as a "3D view-sensitive layout" of controls.

For example, if the user decides to switch to the Top viewpoint, it can be accessed directly in the list. However, on the ViewCube, the Top viewpoint may not be visible and the user would have to select an intermediate viewpoint first to "get to" the Top viewpoint to select it, or they could click and drag to ViewCube to get to the Top view. View selection from a 2D fixed layout has the benefit of being able to go to any view in one click, at the expense of finding the button in a long list and the large amount of screen space that the list consumes. The cost of the 3D view-sensitive cube-based button layout is that it may take up to two clicks to select a viewpoint. However, the cube arrangement has the benefits of relative selection (i.e. it is clear which viewpoints are close to the current viewpoint, which is difficult to determine when presented with a list of viewpoints) and it uses only a small amount of screen space. The extra-click cost may seem high but note that at least 13 –and as many as 19– viewpoints are selectable on the cube at any given time in a single click, depending on its orientation. The remaining 7 to 13 views can be selected quickly, by dragging to rotate the cube around to the other side to the intended viewpoint, or by one additional click (if the first click is used to expose the hidden parts of the cube). In any case, considering object constancy and the principles of direct manipulation, the additional travel of the second animated transition or the freeform rotation would increase overall shape understanding [20].

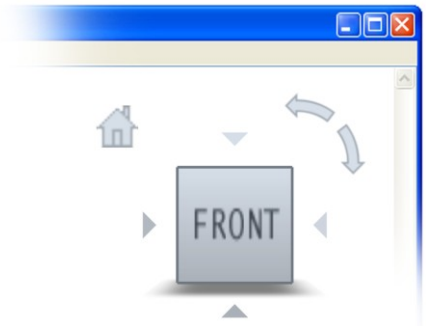

Figure 8. ViewCube Design. Components are the large center cube with selectable faces, edges, and corners, the triangles pointing "around the edges" to orthogonal faces, the "home" button in the top-left to return to a default view, and the "roll" arrows in the top-right for clockwise and counter-clockwise rotation.

3.2 Reaching Orthogonal Views

Switching from one face view, for example "Front", to another, like "Right", is quite common. However, when a ViewCube face is directly facing the user (as in Figure 8), the sides of the ViewCube are not visible, and so, are not directly selectable. If dragging is not used, this leads to the situation where moving from the Front view to the Right view would require two clicks; one click on the Front-Right edge and, once the Right face becomes visible, clicking on the Right face. To remove this extra step, we use the ViewCompass technique. We replaced the cones in the ViewCompass with triangles pointing to the orthogonal faces. Clicking on the triangle on the right side of the ViewCube in Figure 9 would rotate the scene and the cube to the Right view with a single click. Unlike the ViewCompass, these triangles are only visible when directly on a face view. The ViewCube can optionally show face views (or all views) orthographically or in

perspective. However, the ViewCube itself is always rendered in perspective so that edge and corner views of the ViewCube better convey the current viewpoint orientation.

There are also two curved arrows to the top-right of the cube to support clockwise and counter-clockwise view "roll" rotation, which was also identified as a user need in the field of industrial design. Unlike the Glass Cube, the roll arrows are only displayed when available and are presented as separate buttons for clockwise and counter-clockwise actions. As shown in Figure 9, if the user clicks the bottom arrow, the scene and the cube are rotated 90° clockwise. This is especially useful for the top and bottom views where the "correct" orientation of a scene is often highly subjective.

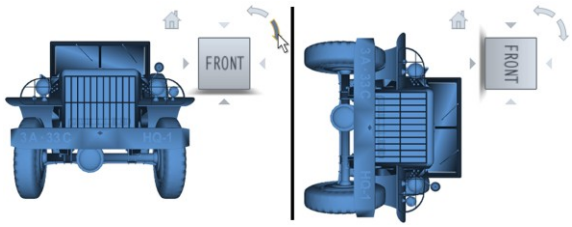

Figure 9. Before (left) and after (right): Clockwise and counter-clockwise controls are shown when at a face view. Clicking the bottom (clockwise) arrow rotates the scene and the cube 90°.

To minimize extra rotations when moving from one viewpoint to another, we use the local up model when moving to the top or bottom faces, and the global up model [Khan et al. 2005] in all other cases.

3.3 Recovery

Finally, to make the viewpoint "recovery" function of the ViewCube more explicit (compared to the implicit ViewCompass approach), we added a "home" icon that is always accessible, shown above and left of the cube in Figure 9. Clicking the Home button returns the view to the standard three-quarter view, or a pre-authored starting view.

A second implicit recovery feature of the ViewCube is available when clicking on a piece of the ViewCube. Once a viewpoint is selected in this manner, while the viewpoint orientation is being changed (during the animated transition), a "fit-to-view" transformation is also applied. This "fit-to-view" function centers the 3D scene (or the selected 3D model if a component of the scene has been selected) ensuring that the (un-shown) bounding box of the scene is completely within the viewing area of the window. This added functionality further helps novice 3D users to recover from navigation problems.

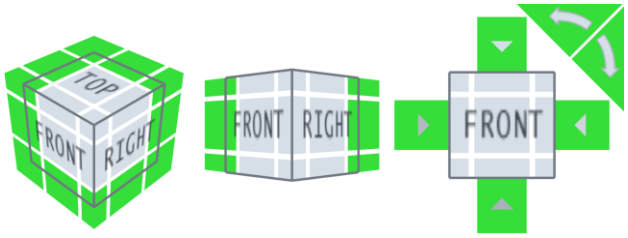

Figure 10. Expanded hit areas for boundary buttons and additional buttons on the ViewCube. Clicking anywhere within the green regions will select the corresponding button.

Finally, while the face buttons are fairly large, the hit areas on the boundary of the cube can become quite small, especially when the view is quite oblique, which occurs when the cube is rotationally close to one of the fixed face views. To help facilitate the selection of the boundary buttons, we algorithmically increase the hit zone areas in these cases (see Figure 10). Interior buttons are the same size as they appear to be but the boundary buttons are more than twice as large as they appear to be. We feel this has been quite effective. During extensive testing, no users expressed that the (visually) smaller ViewCube pieces were difficult to select.

3.4 Rendering

When the view has been rotated a number of times, it becomes more difficult to interpret the current orientation. While the text labels on the cube faces helps, we added some artificial lighting rendering cues such as the drop shadow beneath the cube, as though it was lit from the top, as well as a subtle gradation, from light blue to a darker blue, on the sides of the cube.

Proxy-image labels for viewpoint switching are often used in commercial packages, of either a cube or a model, such as a house, as in Google's SketchUp (see Figure 11). Text lists are also common (see Figure 2a). However, as mentioned earlier, these approaches do not scale well to include all twenty-six views.

Figure 11. View proxy images in Google Sketchup.

So, in addition to the text labels, we implemented designs with 2D proxy labels and no labels. We also created a design containing a proxy model of a 3D house within the cube (see Figure 12).

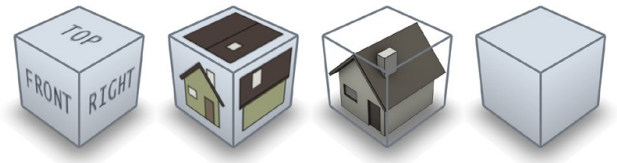

Figure 12. Label Schemes: text, 2D proxy image, 3D proxy model, and no label.

Each labeling scheme and interaction method has advantages and disadvantages as reported in the Experiment section below.

4 Experiment

To help determine the benefits and costs of these design choices, and to better understand how these methods scale from the six view case to the twenty-six views, we conducted a formal experiment.

The goal of the experiment was to measure the performance and subjective preference of the ViewCube design while varying the labeling style and interaction model, clicking vs. dragging.

4.1 Design

The experimental design consisted of three orientation interaction techniques, four labeling schemes and two difficulty levels.

4.1.1 Orientation Interaction Techniques

Specifically, three interaction techniques were studied:

ArcBall – which takes only mouse drag events on the cube performing an ArcBall-style orbit operation with snapping to standard views;

ClickCube – which takes only single mouse click inputs on the ViewCube's faces (and face triangles or roll arrows for orthogonal faces), edges and corners; and

List – which presents an exhaustive collection of buttons to the user similar to a toolbar or items in a pop-up menu (see Figure 13), as well as the roll arrows. This condition represents a much less spatially related arrangement of controls (i.e., "2D fixed layout" for orientation selection) compared to the ClickCube and ArcBall conditions (i.e., "3D view-sensitive layout" for orientation selection).

Both the ClickCube and List conditions animate the view transition to the target view after a ClickCube component (face, edge, or corner) or List button is selected, or any of the additional controls are selected (on mouse-up).

4.1.2 Labeling Schemes

In addition to the three interaction techniques, we varied the labeling schemes to indicate orientation. First, Text labels were used such as "Front" or "Left" to indicate a face view. In the List of buttons condition, combined views were grouped (faces, edges and corners) and labeled with multiple terms such as "Top-Left" or "Front-Top-Right" for an edge and corner view respectively (see Figure 13b). The cube conditions had multiple faces visible to specify combined views (see Figure 12). Next we chose to use 2D Graphics of a familiar object – a house – to indicate orientation. The house is used in numerous 3D applications (see Figure 11) to achieve standard views. The simplified house design consists of unique elements such as a door and chimney to make all faces look unique from each other. For the two cube conditions we defined two additional labeling schemes. The 3D-Model condition embeds an actual 3D model of a miniature house within the cube (see Figure 12) which acts as a proxy for the 3D scene. Finally, we had No Labels on the cube to emphasize relative orientation.

4.1.3 Difficulty Levels

To isolate the effect of easy view orientations (only face views such as "Top") and more challenging orientations (such as "Top-Front-Right"), we defined two levels of target difficulty: Face-Views (the 6 face views) and All-Views (all 26 face, edge and corner views).

4.2 Task – Target Matching

Two shaded models of a car were shown; one in the left hand window (with an initial pseudo-random orientation) and one on the right (shown as the target orientation), as seen in Figure 13a. The subject is asked to orient the car in their (left) window to match the orientation of the car (presented in the right window) using one of the orientation techniques mentioned above. On trials where the target was oriented to the top or the bottom of the car, an additional "roll" rotation step may be needed and the arrow buttons facilitated this action.

In the ArcBall case, the orientations of the target car and subject's car were checked on mouse-up events, and if they matched, a "Trial Success" message was displayed. In the ClickCube and List cases, a mouse-up event initiates playback of an animated transition, after which matching is checked. Each trial would continue until a match was achieved. After a pause of 3 seconds, the next trial would begin. A total of 15 trials were presented for each condition. For the All-Views condition, a subset of 5 faces, 5 edges and 5 corner target orientations were chosen. We ignored the first 3 trials of each condition to factor out subjects becoming familiar with the interface and condition. All subjects experienced the same orientation trials but in random order.

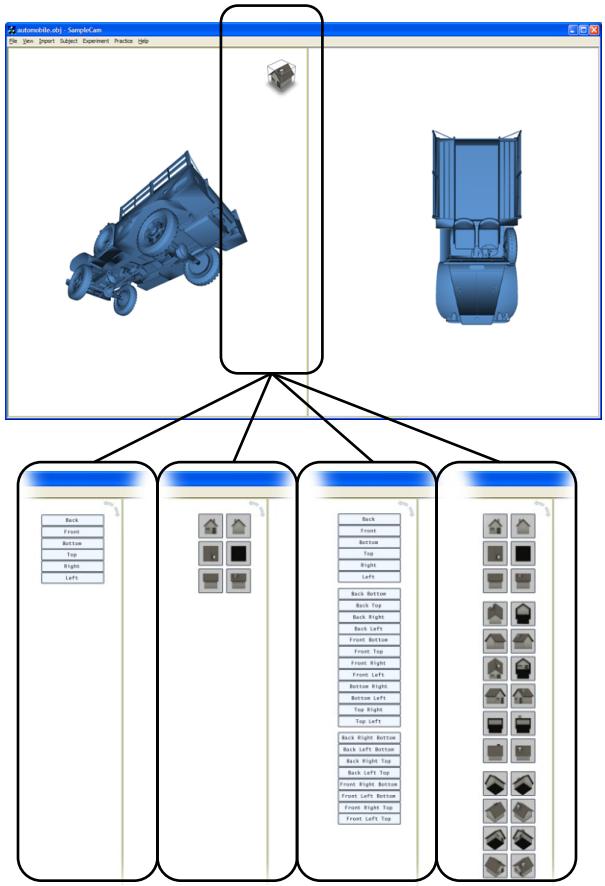

Figure 13. (a) Top: experiment screen. (b) Bottom: the List condition for (left to right) 6 Views with Text Labels, 6 Views with 2D Image Labels, 26 Views with Text Labels, and 26 Views with 2D Image Labels. The area outlined shows how and where the various experiment conditions are displayed.

A total of 18 subjects were used (8 men and 10 women) between the ages of 21 and 29. Roughly half had 3D modeling or extensive video game experience. All were experienced computer users. Before the start of the experiment, we briefly instructed the subjects on how each of the orientation techniques worked. Once comfortable, we asked them to complete the tasks as quickly and as accurately as possible, allowing them to rest between conditions.

A within-subjects design was used with each subject using all three orientation techniques. This resulted in 18 subjects x 3 techniques x 4 labels x 2 target difficulties x 12 trials = 5,184 data points. However, the List condition did not have the 3D-Model or the No Labels labeling conditions, and so this resulted in a total of

4,320 data points for the entire experiment. Trials were grouped by technique and counter balanced with 1/3 of the subjects using the List condition first, 1/3 using the ClickCube first and 1/3 using the ArcBall first. The remaining factors were randomly presented to the subjects.

For every trial, we logged the time for the subject to complete the trial and recorded the number of mouse clicks pressed. Errors were not possible as the system waited for a positive match before proceeding to the next trial. After completing the experiment, subjects were given a short questionnaire to determine their preferences for the three techniques and the four labeling schemes.

5 Results

We performed an analysis of variance on the performance data and found a significant difference between the three techniques (List, ClickCube, and ArcBall) with $F(2, 34) = 74.20$, $p < .0001$. The ArcBall performance was 2.1 times faster than the List technique for the All-Views condition. No statistical significance in performance was found between the List and ClickCube techniques (see Figure 14).

We found a significant difference for task difficulty with $F(1, 17) = 289.73$, $p < .0001$. That is, it took less time to orient to the six face viewpoints compared with the more complicated edge or corner viewpoints for all three techniques.

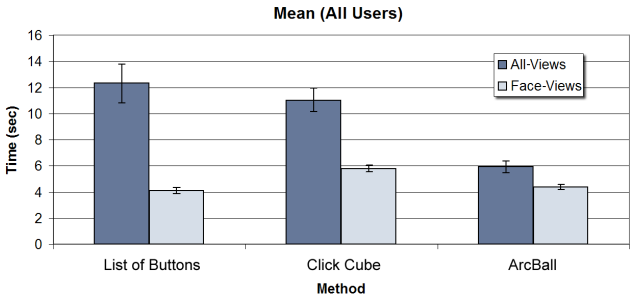

Figure 14. Mean (All Users): List and ClickCube had little difference but the ArcBall was almost twice as fast at the List method.

A significance interaction was found between orientation technique and task difficulty with values of $F(2, 34) = 49.18$, $p < .0001$ (see Figure 14). However, there was no statistically significant performance difference based on the various labeling schemes (Text, 2D-Graphics, 3D-Model and No Labels). A pair-wise T-test of means ($p < .05$) reveals no differences between the labeling schemes within a given orientation technique and task difficulty.

Overall, consistent with the performance results, subjects had a much stronger preference for the ArcBall compared to both the ClickCube and the List of buttons technique. The 3D-Model labeling scheme was preferred over other labeling schemes. Within the List condition, the 2D-Graphics were preferred over the Text labels (see Figure 15).

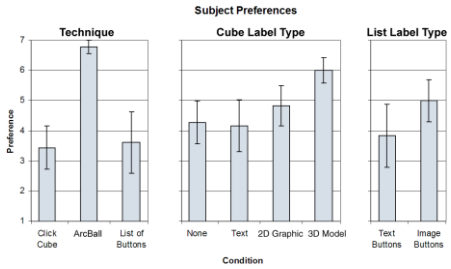

Figure 15. Subject Preferences: the ArcBall and the 3D Model label type were preferred, and the Image buttons were favored over the Text buttons.

6 Discussion

6.1 Orientation Interaction Techniques

The ArcBall greatly outperformed the ClickCube and List techniques. While we expected significant benefits of cube dragging, the factor of 2 times speed increase and the consistency of results were impressive. When using the List method, especially with Text labels, the task seems significantly more difficult than in any other case. As such, it was surprising that the ClickCube had no significant benefits over the List method. We believe this is due to a planning phase of the interaction that is common to both the List and the ClickCube. In practice, before choosing a viewpoint with these two techniques, the user must consider which might be the correct button. Subjects could have used a fast random clicking strategy to complete the trials but we found that, perhaps due to the cost of the animated transitions, they would plan their choices. However, the ArcBall seems to afford immediate operation where the user completely forgoes any planning. If the users finds that they are dragging in the "wrong" direction, they can switch directions very quickly and given the combination of a high C:D ratio and viewpoint snapping, can more easily match the target in a short period of time. It is also interesting that for the simple six viewpoint case, the ArcBall was almost as fast as the short list of the six faces in the List. Furthermore, the ArcBall scales from the six viewpoint case to the twenty-six viewpoint case far better than the other techniques. With the other two techniques, the twenty additional viewpoints grossly affected performance.

Despite the speed advantages of the ArcBall, we believe the ClickCube better supports novice users. In our experience, when users first encounter a new widget, clicking seems to be a more common exploration method than initially dragging on a widget. As such, by supporting both clicking and dragging, we can accommodate both novice and advanced users. Another benefit of the ClickCube operation is the smooth animated transitions to the chosen viewpoint. This action creates a smoother visual flow while working. However, when dragging the ArcBall, the quick model rotation is visually jarring but gives better performance.

6.2 Labeling Schemes

As we expected significant variance based on the labeling scheme, we included four cases in our experiment. In the ArcBall case, it is not terribly surprising to find that label type had little effect because once the subject had clicked on the ViewCube and started dragging, they would invariably switch to visually monitoring the jeep model –not the cube– to match it to the target. However, in the List and ClickCube cases, the subject's performance seems to vary to the individual's talent for

interpreting text, for some people, or imagery for others; but in any case did not significantly vary the time taken to complete the task.

7 Conclusions

Despite the seemingly simple function of a viewpoint orientation indicator and controller, we found many subtleties in designing and evaluating a widget that would accommodate a number of types of users and uses. These factors also play an important role in the overall user experience.

In this paper we presented a novel ViewCube widget as a means for viewpoint switching between 26 views as well as offering 3D orientation information. Our experimental results indicated that users prefer and are twice as fast at using the ViewCube with dragging (to perform orbit operations with view snapping) compared to the "single click based" view switching techniques. Also, by making the scene orientation always visible via the ViewCube as a proxy for the 3D scene, we make visible the effect of other application tools including ArcBall functionality. We found no significant performance effects of varying the ViewCube's labeling schemes (using text, 2D graphics, a 3D proxy model, or no labels) but to facilitate collaboration we believe text labels are most beneficial. In the future, just as the 2D scrollbar facilitates 2D document navigation and orientation, we believe the ViewCube could evolve towards a similar "3D scrollbar" for 3D environments.

Acknowledgments

We thank Anirban Ghosh for originally suggesting the cube design. Also thanks to Ryan Schmidt, Darin Hughes, John Schrag, and Marsha Leverock.

References

BALAKRISHNAN, R. AND KURTENBACH, G. 1999. Exploring bimanual camera control and object manipulation in 3D graphics interfaces. In *Proceedings of ACM CHI*. pp. 56-63.

BAUDISCH, P., CUTRELL, E., HINCKLEY, K., AND EVERSOLE, A. 2005. Snap-and-go: helping users align objects without the modality of traditional snapping. In *ACM Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05*, pp. 301-310.

BIER, E. A. 1990. Snap-dragging in three dimensions. In *ACM Proceedings of the 1990 Symposium on interactive 3D Graphics*, pp. 193-204.

BOWMAN, D., JOHNSON, D. AND HODGES, L. 1999. Testbed environment of virtual environment interaction. In *Proceedings of ACM VRST*. pp. 26-33.

BOWMAN, D., KOLLER, D. AND HODGES, F.H. 1997. Travel in immersive virtual environments. *IEEE VRAIS'97 Virtual Reality Annual International Symposium*. pp. 45-52.

CHAPUIS, O. AND ROUSSEL, N. 2005. Metisse is not a 3D desktop!. In *Proceedings of the 18th Annual ACM Symposium on User interface Software and Technology*, UIST '05, pp. 13-22.

CHITTARO, L. AND BURIGAT, S. 2004. 3D location-pointing as a navigation aid in Virtual Environments. In *ACM Proceedings of the Working Conference on Advanced Visual interfaces* Gallipoli, Italy, May 25 - 28, 2004.. AVI '04, pp. 267-274.

GLEICHER, M. AND WITKIN, A. 1992. Through-the-lens camera control. *ACM SIGGRAPH 92*. pp. 331-340.

GROSSMAN, T., BALAKRISHNAN, R., KURTENBACH, G., FITZMAURICE, G., KHAN, A., AND BUXTON, B. 2001. Interaction techniques for 3D modeling on large displays. In *Proceedings of the 2001 Symposium on interactive 3D Graphics* SI3D '01, pp. 17-23.

HANSON, A. AND WERNET, E. 1997. Constrained 3D navigation with 2D controllers. *IEEE Visulization*. pp. 175-182.

HENDERSON, D. A. AND CARD, S. 1986. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Transactions on Graphics TOG., 53., pp. 211–243.

IGARASHI, T., KADOBAYASHI, R., MASE, K. AND TANAKA, H. 1998. Path drawing for 3D walkthrough. *ACM UIST*. pp. 173-174.

JUL, S., & FURNAS, G. 1998. Critical zones in desert fog: aids to multiscale navigation. *ACM Symposium on User Interface Software and Technology*. pp. 97-106.

KHAN, A. KOMALO, B., STAM, J., FITZMAURICE, G. AND KURTENBACH, G. 2005. HoverCam: interactive 3D navigation for proximal object inspection, In *Proceedings of ACM Symposium on Interactive 3D graphics and games*, pp. 73-80.

KOMERSKA, R. AND WARE, C. 2003. "Haptic-GeoZui3D: Exploring the Use of Haptics in AUV Path Planning", *Proceedings 13th International Symposium on Unmanned Untethered Submersible Technology UUST'03.*, Durham, NH.

MACKINLAY, J., CARD, S. AND ROBERTSON, G. 1990. Rapid controlled movement through a virtual 3D workspace. *ACM SIGGRAPH 90*. pp. 171-176.

PHILLIPS, C.B., BADLER, N.I., AND GRANIERI, J. 1992. Automatic Viewing Control for 3D Direct Manipulation. In *ACM Proceedings of ACM Symposium on Interactive 3D Graphics*, pp. 71-74.

PIERCE, J. S., CONWAY, M., VAN DANTZICH, M., AND ROBERTSON, G. 1999. Toolspaces and glances: storing, accessing, and retrieving objects in 3D desktop applications. In *ACM Proceedings of the 1999 Symposium on interactive 3D Graphics*, SI3D, pp. 163-168.

REZZONICO, S. AND THALMANN, D. 1996. Browsing 3D Bookmarks in BED, WebNet 1996.

ROBERTSON, G., CARD, S.K., AND MACKINLAY, J.D. 1989. The cognitive coprocessor architecture for interactive user interfaces, In *Proceedings of ACM UIST 1989*, pp 10-18.

SHNEIDERMAN, B. 1983.. Direct manipulation: A step beyond programming languages. *IEEE Computer, 168.*, pp 57-69.

SHOEMAKE, K. 1992. Arcball: A User Interface for Specifying Three-Dimensional Orientation Using a Mouse, *Proc. Graphics Interface '92*, pp. 151-156.

SINGH, K. AND BALAKRISHNAN, R. 2004. Visualizing 3D Scenes using Non-Linear Projections and Data Mining of Previous Camera Moves. In *Proceedings of ACM Computer Graphics, Virtual Reality, Visualization and Interaction in Africa*, pp. 41-48.

SMITH, G., SALZMAN, T., & STUERZLINGER, W. 2001. 3D Scene Manipulation with 2D Devices and Constraints. *Proceedings of Graphics Interface*, pp. 135-142.

STEED, A. 1997. Efficient navigation around complex virtual environments. *ACM VRST*. pp. 173-180.

STOAKLEY, R., CONWAY, M. AND PAUSCH, R. 1995. Virtual reality on a WIM: Interactive worlds in miniature. *ACM CHI*. pp. 265-272.

STOEV, S.L. AND STRABER, W. 2002. A Case Study on Automatic Camera Placement and Motion for Visualizing Historical Data. *Proceedings of IEEE Visualization*, pp. 545-558.

TAN, D., ROBERTSON, G. AND CZERWINSKI, M. 2001. Exploring 3D navigation: combining speed-coupled flying with orbiting. *ACM CHI*. pp. 418-425.

TORY, M. 2003. Mental Registration of 2D and 3D Visualizations An Empirical Study.. In *Proceedings of the 14th IEEE Visualization 2003 Vis'03*. pp. 371-379.

TORY, M. AND SWINDELLS, C. 2003. Comparing ExoVis, Orientation Icon, and In-Place 3D Visualization Techniques. In *Proceedings of Graphics Interface*, pp. 57-64.

WARE, C. AND FLEET, D. 1997. Context sensitive flying interface. *ACM Symposium on Interactive 3D Graphics*. pp. 127-130.

WARE, C. AND OSBORNE, S. 1990. Exploration and virtual camera control in virtual three dimensional environments. *ACM Symposium on Interactive 3D Graphics*. pp. 175-183.

ZELEZNIK, R. AND FORSBERG, A. 1999. UniCam - 2D Gestural Camera Controls for 3D Environments. *ACM Symposium on Interactive 3D Graphics*. 169-173.